NumPy 和陣列導向的程式設計

NumPy 可以說是 Python 中最最標準的科學計算、數據分析套件。也因為 NumPy 的出現, 讓 Python 有了非常好的數據分析基礎, 一直到現在成為數據分析霸主。

In [2]:
```
import numpy as np
```

1. 陣列導向 101

科學計算一個很核心的概念叫 "array oriented" 的寫法。Array 是 numpy 標準的資料結構, 和 list 很像, 但就差了那麼一點點。而這一點點讓我們在計算上是無比的方便。

【暖身】 計算平均

某位同學期中考各科成績如下, 請幫他計算成績。

```
grades = [77, 85, 56, 90, 66]
```

請計算平均。

In [7]:
```
s = 0
grades = [77, 85, 56, 90, 66]
for i in grades:
    s = s + i
print("AVG={0:4.2f}".format(s/len(grades)))
```

```
AVG=74.80
```

【示範】陣列導向

In [10]:
```
s = 0
grades = [77, 85, 56, 90, 66]
arr = np.array(grades)
arr.mean()
```

Out[10]:
```
74.8
```

最大值

In [11]:
```
arr.max()
```

Out[11]:
```
90
```

標準差

In [12]:
```
arr.std()
```

Out[12]:
```
12.416118556135006
```

【暖身】 換算匯率

假設今天我想查查號稱 Pentax 三公主的 31mm, 43mm, 77mm 三隻 limited 鏡頭在美國賣多少。於是我去 B&H 查了他們的價格分別是:

```
prices = [1096.95, 596.95, 896.95]
```

我又查了 Google 匯率 1 美金為 31.71 元。請把三支鏡頭的價格換算為台幣。

In [23]:
```
prices = [1096.95, 596.95, 896.95]
usd2twd = 31.71
twd_proces = []
for i in prices:
    twd_proces.append(i * usd2twd)
print(twd_proces)
```

```
[34784.2845, 18929.2845, 28442.2845]
```

先不管這實在有夠醜的數字, 我們要記得在科學計算中:

儘可能不要使用迴圈

這可能嗎?

【示範】陣列換算匯率

In [14]:
```
usd2twd = 31.71
prices = [1096.95, 596.95, 896.95]
prices = np.array(prices)
prices * usd2twd
```

Out[14]:
```
array([34784.2845, 18929.2845, 28442.2845])
```

哦哦, 傑克, 這太神奇了!

2. 其實 array 還有很多功能

【練習】 成績計算

一位老師成績這樣算的:

- 平時成績 20%
- 期中考 35%
- 期末考 45%

有位同學

- 平時成績 85 分
- 期中 70 分
- 期末 80 分

這位同學的學期成績是多少?

In [24]:
```
grades = np.array([85,70,80])
weights = np.array([0.2, 0.35, 0.45])
```

這還不是我們要的最終成績啊!

In [25]:
```
grades * weights
```

Out[25]:
```
array([17. , 24.5, 36. ])
```

【提示】 array 還有很多函數可以用

可以先打入

```
weighted_grades.
```

先不要按 `enter` 或 `shift-enter`，而是按 `tab` ...

In [27]:
```
weighted_grades = grades * weights
weighted_grades.sum()
```

Out[27]:
```
77.5
```

【技巧】 一行完成成績計算

In [29]:
```
np.dot(grades,weights)
```

Out[29]:
```
77.5
```

3. 重要的 array 大變身!

我們在數據分析, 常常要改 array 的型式。

【練習】 一個 50 個數字的 array

先想辦法、用亂數做出 50 個數字的 array, 叫做 A 好了。

In [6]:
```
A = np.random.rand(50)
A
```

Out[6]:
```
array([0.71832962, 0.28669108, 0.47543355, 0.77350425, 0.31416229,
       0.17130915, 0.52439379, 0.13721938, 0.56711909, 0.03041177,
       0.5244325 , 0.1531325 , 0.23387609, 0.03047107, 0.83923267,
       0.3093713 , 0.31664687, 0.11353862, 0.18870403, 0.06721235,
       0.83329003, 0.87768132, 0.65839295, 0.98824787, 0.39985083,
       0.91339093, 0.33686305, 0.88456216, 0.79834542, 0.39299245,
       0.03283734, 0.21470191, 0.75236508, 0.57115821, 0.45215219,
       0.71419333, 0.62730833, 0.43588986, 0.60730637, 0.774242  ,
       0.30798526, 0.70510129, 0.41574119, 0.07918249, 0.0427535 ,
       0.9807242 , 0.40744182, 0.94332834, 0.83713745, 0.63318054])
```

【技巧】 檢查 A 的 shape

In [5]:
```
A.shape
```

Out[5]:
```
(50,)
```

【技巧】 更改 A 的 shape

In [17]:
```
A.shape = (25, 2)
A[0,0]
```

Out[17]:
```
0.7183296197742031
```

【技巧】 也可以用 reshape

但要注意, reshape 並沒有改原來的陣列。

In [25]:
```
A.reshape(10, 5)
A
```

Out[25]:
```
array([[0.71832962, 0.28669108],
       [0.47543355, 0.77350425],
       [0.31416229, 0.17130915],
       [0.52439379, 0.13721938],
       [0.56711909, 0.03041177],
       [0.5244325 , 0.1531325 ],
       [0.23387609, 0.03047107],
       [0.83923267, 0.3093713 ],
       [0.31664687, 0.11353862],
       [0.18870403, 0.06721235],
       [0.83329003, 0.87768132],
       [0.65839295, 0.98824787],
       [0.39985083, 0.91339093],
       [0.33686305, 0.88456216],
       [0.79834542, 0.39299245],
       [0.03283734, 0.21470191],
       [0.75236508, 0.57115821],
       [0.45215219, 0.71419333],
       [0.62730833, 0.43588986],
       [0.60730637, 0.774242  ],
       [0.30798526, 0.70510129],
       [0.41574119, 0.07918249],
       [0.0427535 , 0.9807242 ],
       [0.40744182, 0.94332834],
       [0.83713745, 0.63318054]])
```

【技巧】 拉平 ravel

雖然你想一想就知道可以用 `shape` 或 `reshape` 把多維陣列拉成一維。不過用 `ravel` 很潮。

In [26]:
```
A.ravel()
```

Out[26]:
```
array([0.71832962, 0.28669108, 0.47543355, 0.77350425, 0.31416229,
       0.17130915, 0.52439379, 0.13721938, 0.56711909, 0.03041177,
       0.5244325 , 0.1531325 , 0.23387609, 0.03047107, 0.83923267,
       0.3093713 , 0.31664687, 0.11353862, 0.18870403, 0.06721235,
       0.83329003, 0.87768132, 0.65839295, 0.98824787, 0.39985083,
       0.91339093, 0.33686305, 0.88456216, 0.79834542, 0.39299245,
       0.03283734, 0.21470191, 0.75236508, 0.57115821, 0.45215219,
       0.71419333, 0.62730833, 0.43588986, 0.60730637, 0.774242  ,
       0.30798526, 0.70510129, 0.41574119, 0.07918249, 0.0427535 ,
       0.9807242 , 0.40744182, 0.94332834, 0.83713745, 0.63318054])
```

4. 快速 array 生成法

【技巧】 都是 0 的 array

In [27]:
```
np.zeros(10)
```

Out[27]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

【技巧】 都是 1 的 array

In [28]:
```
np.ones(10)
```

Out[28]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

【技巧】單位矩陣

In [30]:
```
np.eye(5)
```

Out[30]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

【技巧】給定範圍均勻生出 n 個點

In [33]:
```
x = np.linspace(0,10,100) # 0~10範圍 ,切100個點
x
```

Out[33]:
```
array([ 0.        ,  0.1010101 ,  0.2020202 ,  0.3030303 ,  0.4040404 ,
        0.50505051,  0.60606061,  0.70707071,  0.80808081,  0.90909091,
        1.01010101,  1.11111111,  1.21212121,  1.31313131,  1.41414141,
        1.51515152,  1.61616162,  1.71717172,  1.81818182,  1.91919192,
        2.02020202,  2.12121212,  2.22222222,  2.32323232,  2.42424242,
        2.52525253,  2.62626263,  2.72727273,  2.82828283,  2.92929293,
        3.03030303,  3.13131313,  3.23232323,  3.33333333,  3.43434343,
        3.53535354,  3.63636364,  3.73737374,  3.83838384,  3.93939394,
        4.04040404,  4.14141414,  4.24242424,  4.34343434,  4.44444444,
        4.54545455,  4.64646465,  4.74747475,  4.84848485,  4.94949495,
        5.05050505,  5.15151515,  5.25252525,  5.35353535,  5.45454545,
        5.55555556,  5.65656566,  5.75757576,  5.85858586,  5.95959596,
        6.06060606,  6.16161616,  6.26262626,  6.36363636,  6.46464646,
        6.56565657,  6.66666667,  6.76767677,  6.86868687,  6.96969697,
        7.07070707,  7.17171717,  7.27272727,  7.37373737,  7.47474747,
        7.57575758,  7.67676768,  7.77777778,  7.87878788,  7.97979798,
        8.08080808,  8.18181818,  8.28282828,  8.38383838,  8.48484848,
        8.58585859,  8.68686869,  8.78787879,  8.88888889,  8.98989899,
        9.09090909,  9.19191919,  9.29292929,  9.39393939,  9.49494949,
        9.5959596 ,  9.6969697 ,  9.7979798 ,  9.8989899 , 10.        ])
```

【技巧】range 的 array 版

就是 arange 。

In [34]:
```
np.arange(1,10)
```

Out[34]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

5. 超重要 axis 觀念

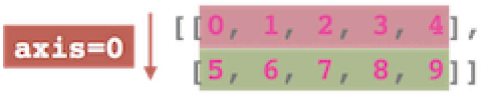
初學 numpy 很多人有點弄不清楚 axis 概念。其實掌握矩陣, 或很像矩陣的陣列都是「先列後行」就可以!

我們先弄個 array 來練習。

In [43]:
```
A = np.arange(10).reshape(2, 5)
A
```

Out[43]:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

【重點】 一列一列算下來是 axis=0



In [44]:
```
A.sum(axis=0)
```

Out[44]:
```
array([ 5,  7,  9, 11, 13])
```

【重點】 一行一行算過去是 axis=1

In [48]:
```
A.sum(axis=1)
```

Out[48]:
```
array([10, 35])
```

【提示】當然也有可能全部算

In [54]:
```
A.sum()
```

Out[54]:
```
45
```

6. array 過濾器

篩出我們要的資料, 這樣的技巧非常重要!

【例子】篩出大於 0 的數

我們有個陣列, 想找出大於 0 的數。
L = np.array([3, -2, -1, 5, 7, -3])

In [92]:
```
L = np.array([3, -2, -1, 5, 7, -3])
L[L>0]
```

Out[92]:
```
array([3, 5, 7])
```

我們可以很白痴的自己判斷...
c = np.array([True,False,False,True,True,False])

In [52]:
```
c = np.array([True,False,False,True,True,False])
```

這是做啥呢? 我們可以瞬間...

In [53]:
```
L[c]
```

Out[53]:
```
array([3, 5, 7])
```

除了自己做很白痴, 這看來很厲害!

事實上我們可以叫 `numpy` 做!

In [56]:
```
L>0
```

Out[56]:
```
array([ True, False, False,  True,  True, False])
```

這有點強, 我們還可以一次到位!

In [57]:
```
L[L>0]
```

Out[57]:
```
array([3, 5, 7])
```

7. 次元切割刀

`numpy` 中 array 的切割法和 list 很像。

In [58]:
```
x = np.arange(10)
x
```

Out[58]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [63]:
```
x[3:7]
```

Out[63]:
```
array([3, 4, 5, 6])
```

【技巧】2維陣列切法

記得先列後行!

In [67]:
```
x.shape=(2,5)
x
```

Out[67]:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

要所有的row, 切出行 1-3 位置。

In [73]:
```
x[:,1:3]
```

Out[73]:
```
array([[1, 2],
       [6, 7]])
```

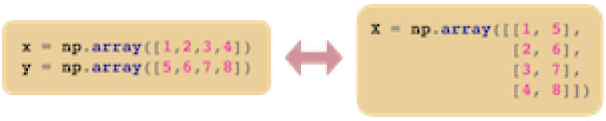
要所有的行col, 切出第 1 列row!

In [71]:
```
x[1,:]
```

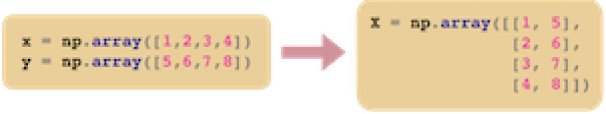
Out[71]:
```
array([5, 6, 7, 8, 9])
```

8. NumPy 的 zip 和 unzip

之前我們介紹 list 可以用 `zip` 和 `unzip` (其實還是 `zip`) 做到的資料格式變換, 在 array 中怎麼做呢?



【重點】array 的 zip



In [75]:
```
x = np.array([1,2,3,4])
y = np.array([5,6,7,8])
```

In [89]:
```
ans=np.c_[x,y] #col
ans
```

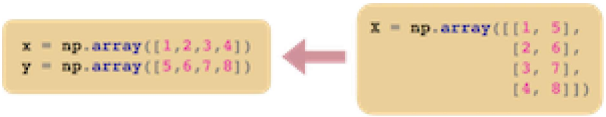
Out[89]:
```
array([[1, 5],
       [2, 6],
       [3, 7],
       [4, 8]])
```

In [87]:
```
ans=np.r_[x,y] #row
ans
```

Out[87]:
```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

【重點】array 的 unzip

這裡其實只需要用到 array 的切割法...



In [90]:
```
ans[:,0]
```

Out[90]:
```
array([1, 2, 3, 4])
```

In [86]:
```
ans[:,1]
```

Out[86]:
```
array([5, 6, 7, 8])
```